A. CAHIER DES CHARGES DU TP3- PROGRAMMATION EN ASSEMBLEUR DU MICROCONTOLEUR MC68HC11

I. Objectifs :

- ✔ Utiliser l'environnement du développement « MiniIDE » et le simulateur « Wookie » pour développer des programmes du microcontrôleur 68HC11.
- ✔ Utiliser et commander les sorties parallèles du 68HC11 (Port B) pour l'allumage séquentiel d'une ligne de LEDs (L0 à L7).
- ✔ Générer un signal carré, de rapport cyclique 50%, sur un bit d'un port du 68HC11 et signal carré de rapport cyclique 20% à l'aide du TIMER ;
- ✔ Ecrire un programme qui copie un octet entré sur le Port C et qui l'affiche sur le Port B.

II. Travaux réalisés :

1. Premier Programme :

Réalisation d'un programme qui commence par allumer la premier LED L0 (connecté au bit 0 du port B) et à la séquence suivante, la LED L1 (connecté au bit 1 du port B), et ainsi de suite (Sens d'allumage dirigé de la droite vers la gauche , séquence : L0,L1,…,L7) .Quand la dernier LED L7 est allumée on repart dans l'autre sens (sens d'allumage dirigé de la gauche vers la droite, séquence: (L7,L6 ,L5, ….L0), et on recommence indéfiniment.

1.1. Ecriture et compilation du programme :

On saisit le programme dans l'environnement de développement « MiniIDE », puis on l'enregistre sous l'extension .asm 'Programme LED.asm'.

Remarque :

- Si le programme s'écrit en couleurs, ça veut dire que l'écriture est correcte, par contre s'il est tout en noir, il existe une erreur quelque part.
- On peut ajouter des commentaires après chaque ligne de programme (; commentaire)

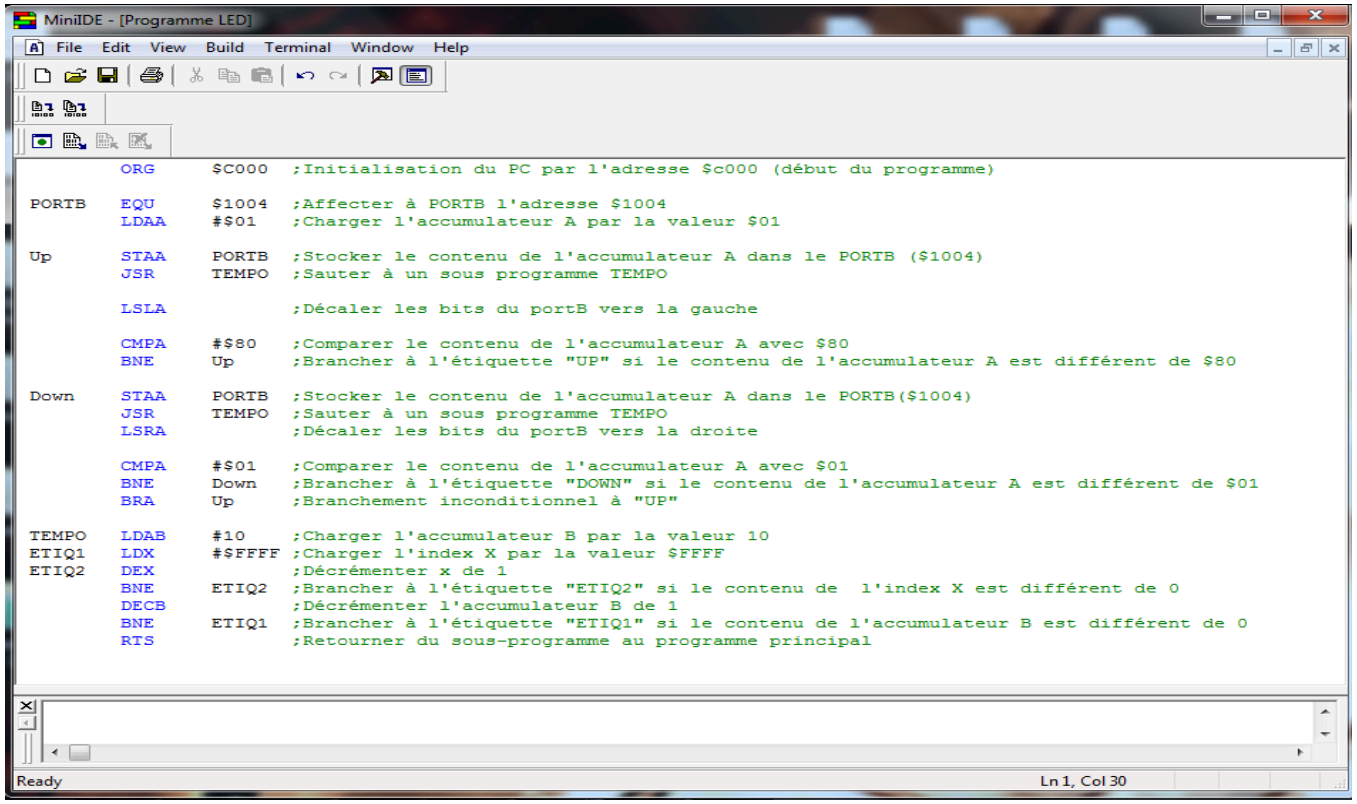


Image1 : Programme LED.asm

- ✔ On compile le programme avec 'Build Current'
- ✔ Un message s'affiche dans la fenêtre 'Output Window' indiquant l'emplacement du programme, nombre d'avertissement et d'erreurs.

Une compilation réussite produit deux fichiers :

- ✔ Programme LED.lst
- ✔ Programme LED. S19

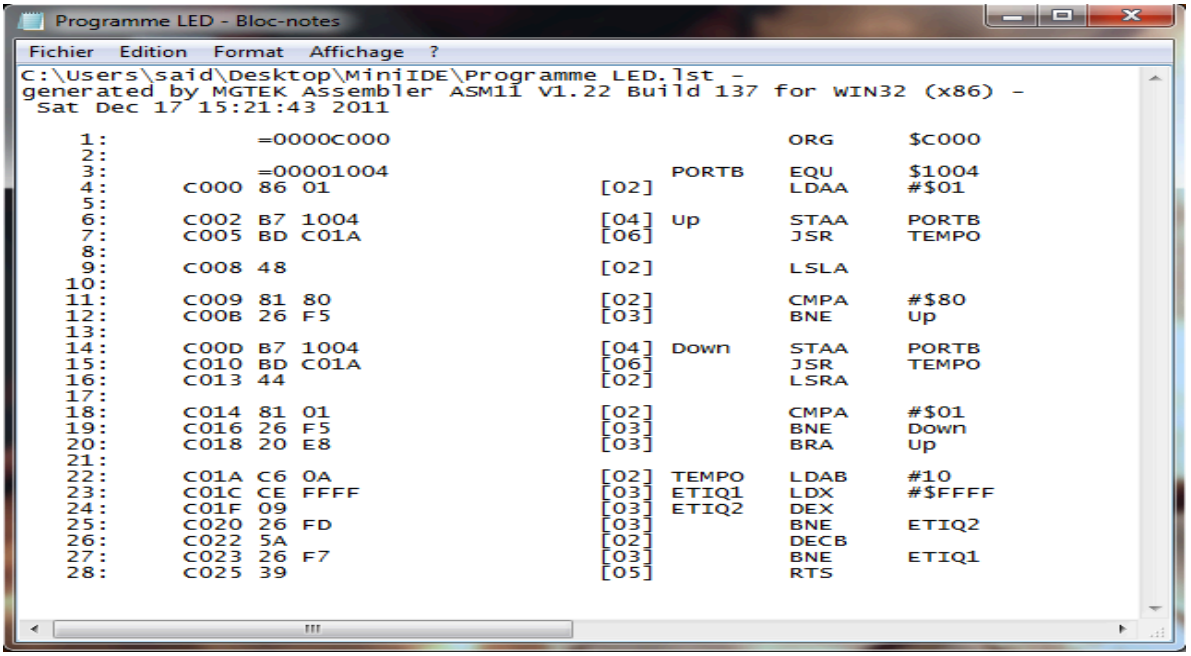


Image2 : Programme LED.lst

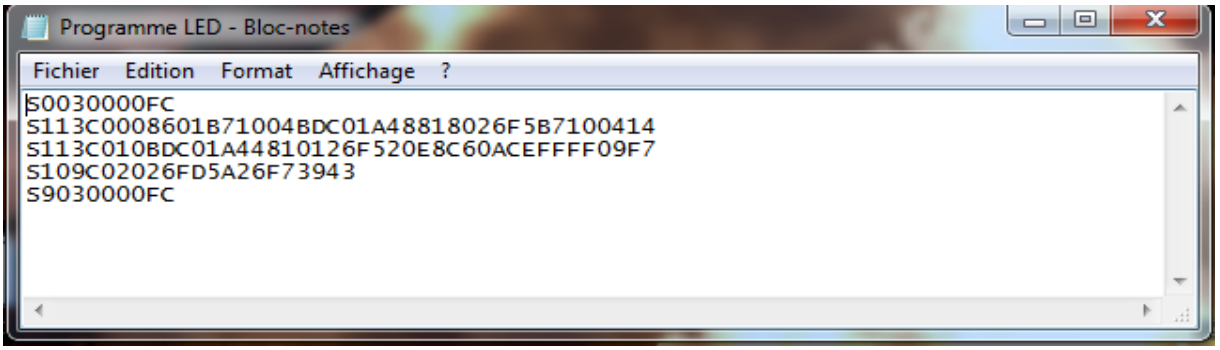


Image3 : Programme LED.S19

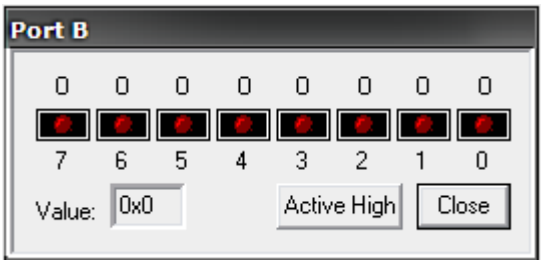
On est maintenant prêt à charger et exécuter notre programme.

2.2. Chargement et exécution du programme :

On lance le simulateur « Wookie »

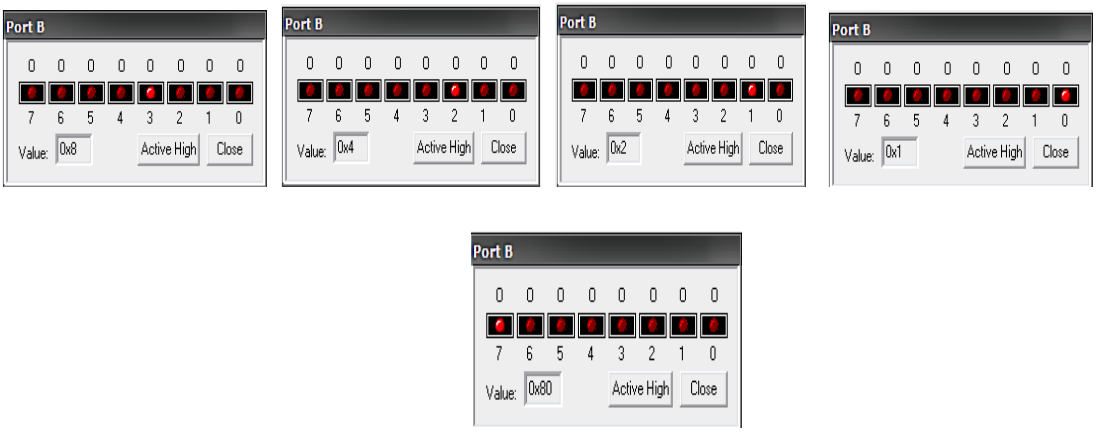
- ✔ On charge le fichier Programme LED.S19 avec ‘Load .S19 file’ dans le menu ‘File’.
- ✔ Dans la fenêtre ‘Set HC11 Mode’ on saisit l’adresse de début de programme ($C000)
- ✔ En suite une fenêtre ‘Choose LST file format’, on choisit ‘GCC3 [Addr : Code] et on tape OK.
- ✔ Une fenêtre ‘Code View’ s’affiche qui contient le fichier .lst du programme.

Avant l’exécution :

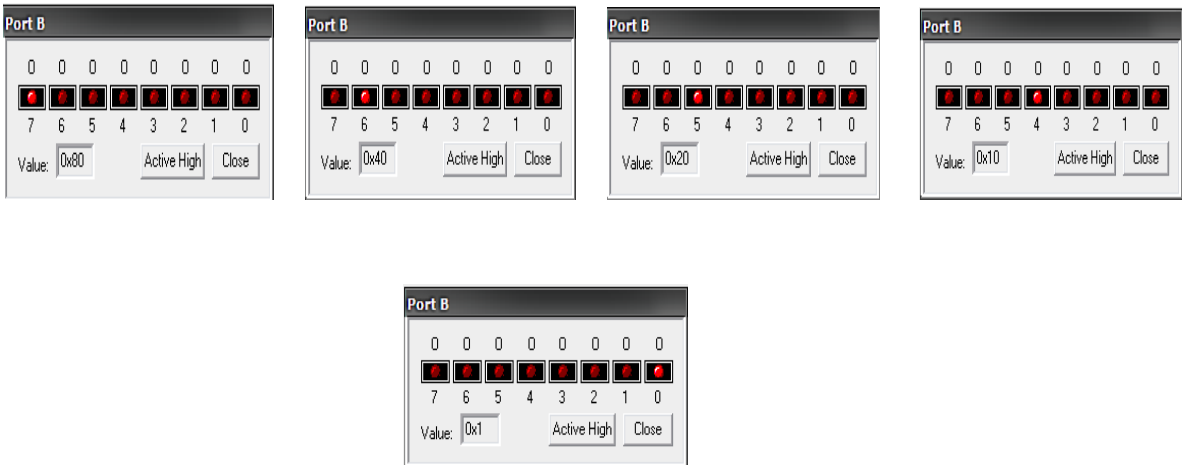


Après l'exécution

- *Sens d'allumage dirigé de la droite vers la gauche, séquence : L0, L1, L2, L3…, L7*



- *Sens d'allumage dirigé de la gauche vers la droite, séquence : L7, L6, L5, L4…, L0*



Le programme s'est bien effectué.

2. Deuxième Programme :

2 *Premier sous programme:* *signal carré Sans utiliser le TIMER α=50*

a) Ecriture et compilation de programme

On saisit le programme dans l'environnement de développement « MiniIDE », puis on l'enregistre sous l'extension .asm 'signal carré Sans utiliser le TIMER α=50.asm'.

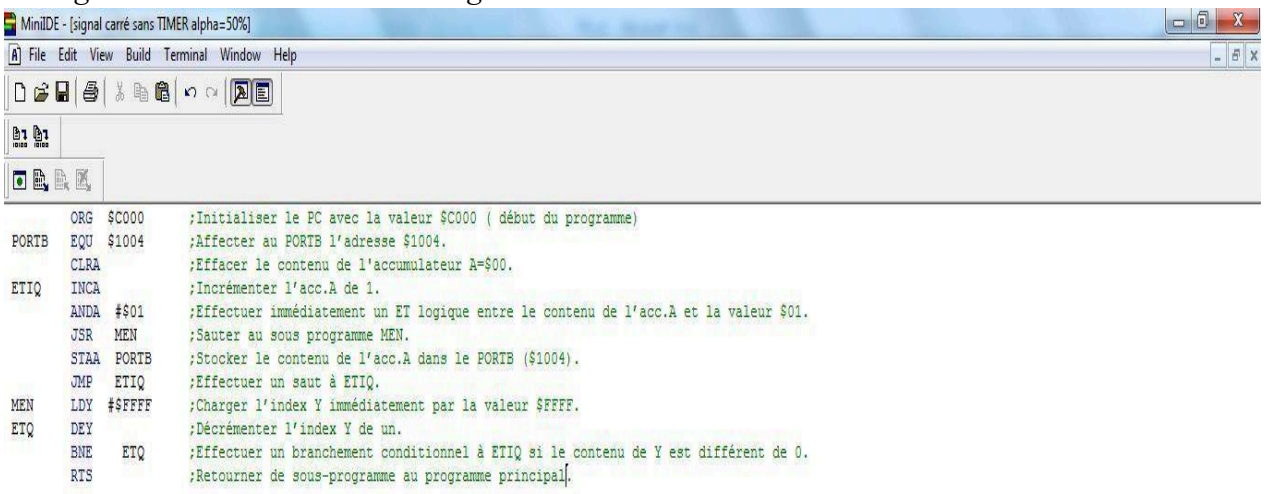


Image1 : Programme LED.asm

- ✔ On compile le programme avec 'Build Current'
- ✔ Un message s'affiche dans la fenêtre 'Output Window' indiquant l'emplacement du programme, nombre d'avertissement et d'erreurs.

Une compilation réussite produit deux fichiers :

- ✔ *signal carré Sans utiliser le TIMER α=50*.lst
- ✔ *signal carré Sans utiliser le TIMER α=50*. S19

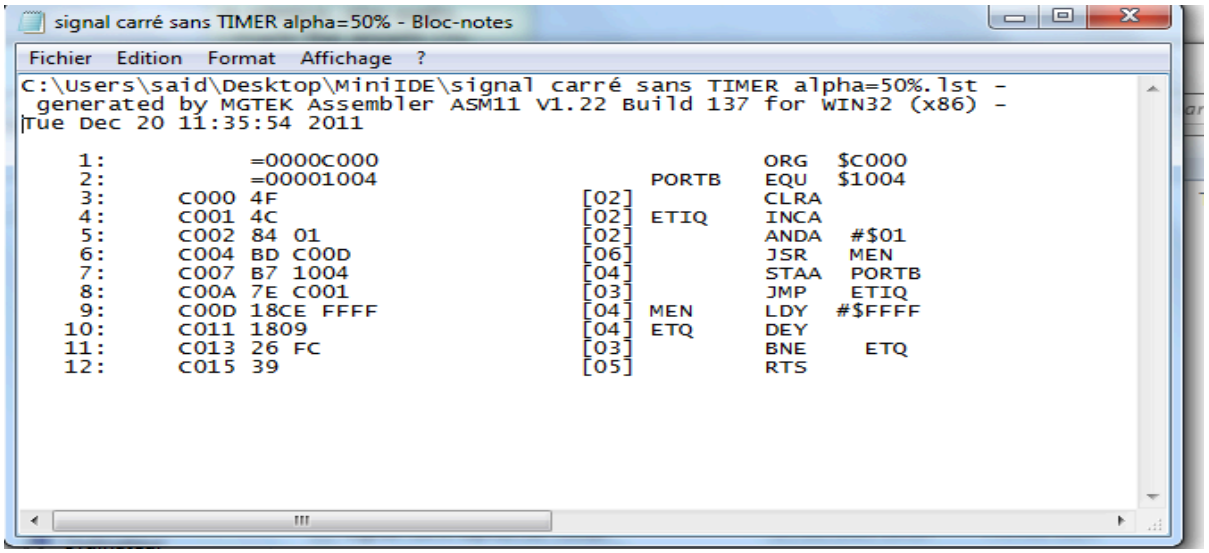


Image2 : *signal carré Sans utiliser le TIMER α=50*.lst

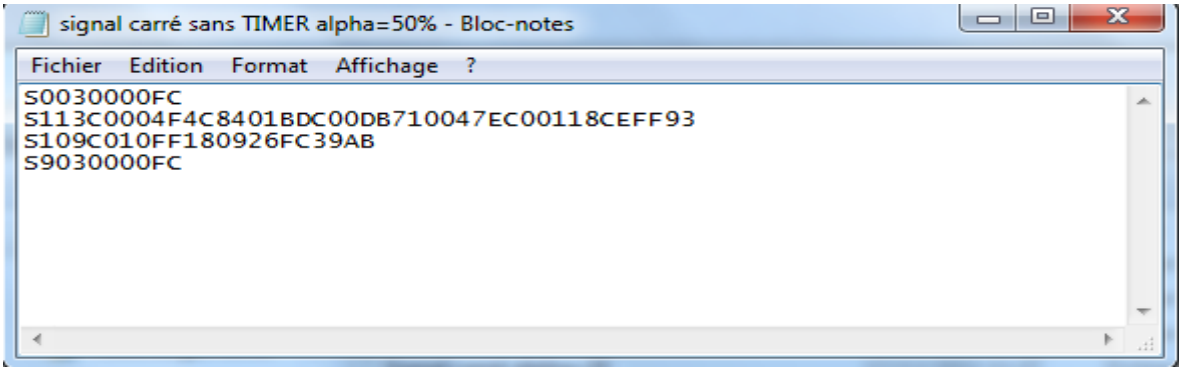


Image3 : *signal carré Sans utiliser le TIMER α=50*.S19

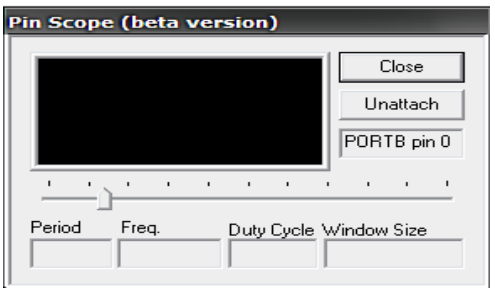
On est maintenant prêt à charger et exécuter notre programme.

2.2. Chargement et exécution du programme :

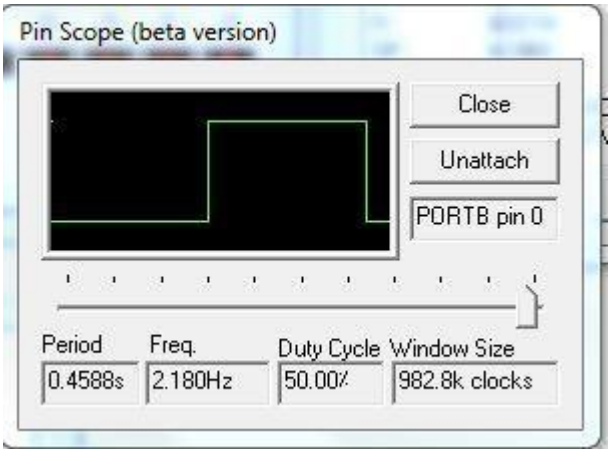
On lance le simulateur « Wookie »

- ✔ On charge le fichier *signal carré Sans utiliser le TIMER α=50*.S19 avec 'Load .S19 file' dans le menu 'File'.
- ✔ Dans la fenêtre Set HC11 Mode' on saisit l'adresse de début de programme ($C000)
- ✔ En suite une fenêtre 'Choose LST file format', on choisit 'GCC3 [Addr : Code] et on tape OK.
- ✔ Une fenêtre 'Code View' s'affiche qui contient le fichier .lst du programme.

Avant l'exécution :



Après l'exécution :



Après l'exécution on voit bien que notre programme est bien exécuté.
D'après la visualisation du signal on voit que sa fréquence est de **2.180Hz.**

2 *Deuxième sous programme: signal carré avec utilisation du TIMER α=50*

2.1. *Ecriture et compilation du programme :*

✔ *On saisit le programme dans l'environnement de développement « MiniIDE », puis on l'enregistre sous l'extension .asm 'signal carré avec Timer alpha=50.asm'.*

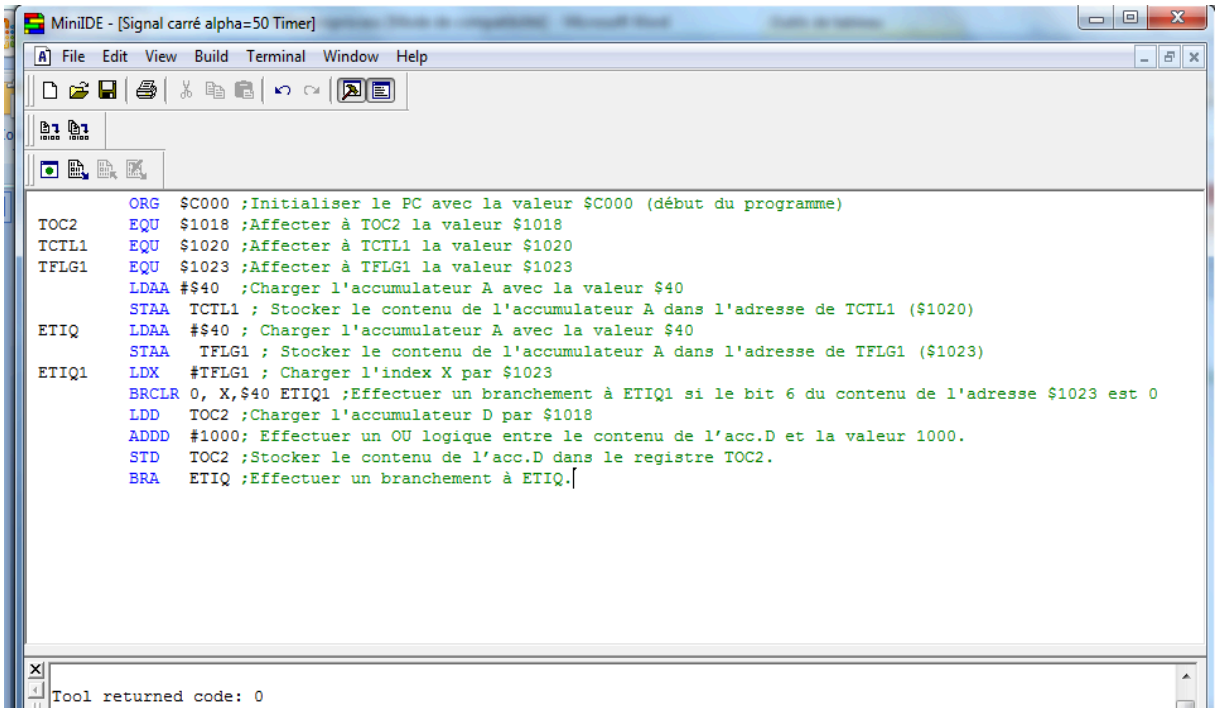


Image1 : *signal carré avec TIMER α=50*.asm

✔ On compile le programme avec 'Build Current'

✔ Un message s'affiche dans la fenêtre 'Output Window' indiquant l'emplacement du programme, nombre d'avertissement et d'erreurs.

Une compilation réussite produit deux fichiers :

✔ *signal carré avec TIMER α=50*.lst

✔ *signal carré avec le TIMER α=50*. S19

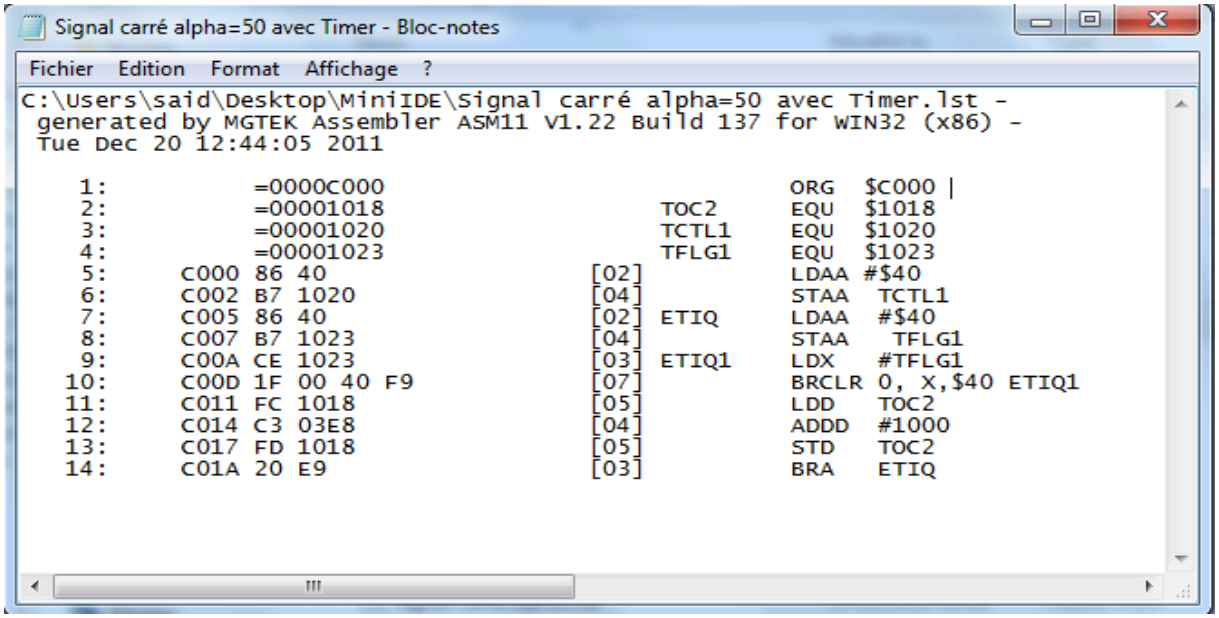


Image2 : *signal carré avec TIMER α=50*.lst

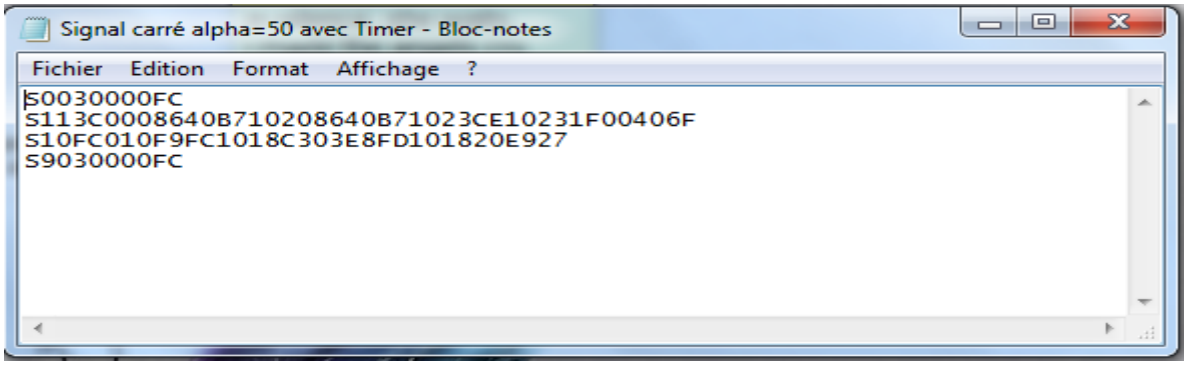


Image3 : *signal carré avec TIMER α=50*.S19

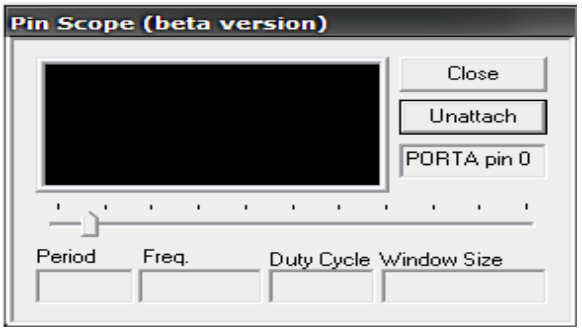
On est maintenant prêt à charger et exécuter notre programme.

2.2. Chargement et exécution du programme :

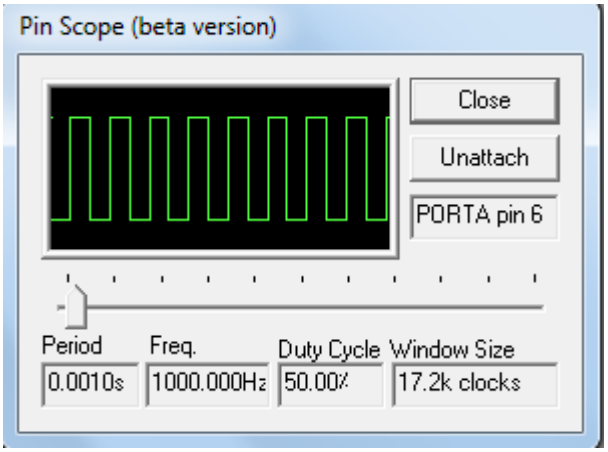
On lance le simulateur « Wookie »

- ✔ On charge le fichier *signal carré avec TIMER α=50*.S19 avec 'Load .S19 file' dans le menu 'File'.
- ✔ Dans la fenêtre 'Set HC11 Mode' on saisit l'adresse de début de programme ($C000)
- ✔ En suite une fenêtre 'Choose LST file format', on choisit 'GCC3 [Addr : Code] et on tape OK.
- ✔ Une fenêtre 'Code View' s'affiche qui contient le fichier .lst du programme.

Avant l'exécution :



Après l'exécution :



Après l'exécution on voit bien que notre programme est bien exécuté.
D'après la visualisation du signal on voit que sa fréquence est de **1000Hz.**

3 *Troisième sous programme: signal carré avec utilisation du TIMER α=20*

2.2. *Ecriture et compilation du programme :*

- ✔ *On saisit le programme dans l'environnement de développement « MiniIDE », puis on l'enregistre sous l'extension .asm 'signal carré avec Timer alpha=20.asm'.*

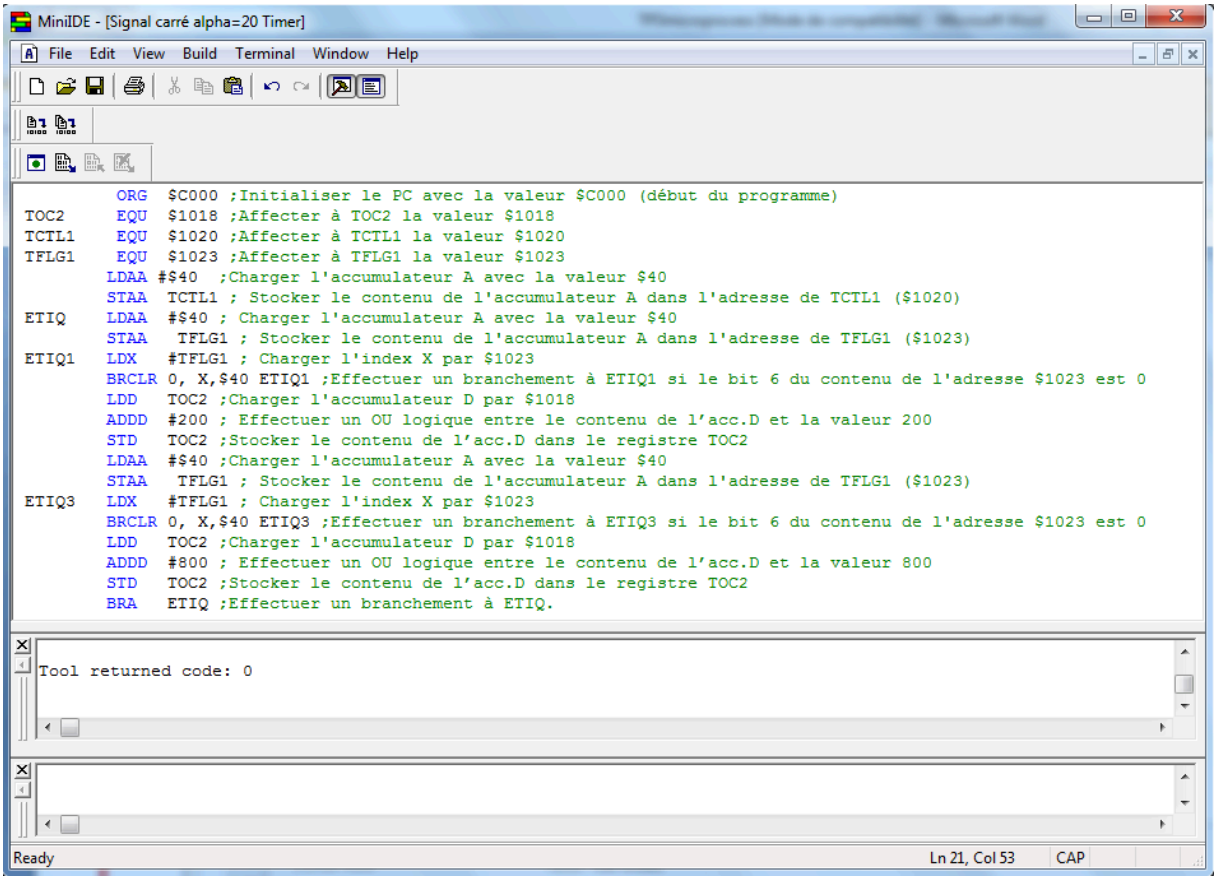


Image1 : *signal carré avec TIMER α=20*.asm

✔ On compile le programme avec 'Build Current'

✔ Un message s'affiche dans la fenêtre 'Output Window' indiquant l'emplacement du programme, nombre d'avertissement et d'erreurs.

Une compilation réussite produit deux fichiers :

✔ *signal carré avec TIMER α=20*.lst

✔ *signal carré avec le TIMER α=20*. S19

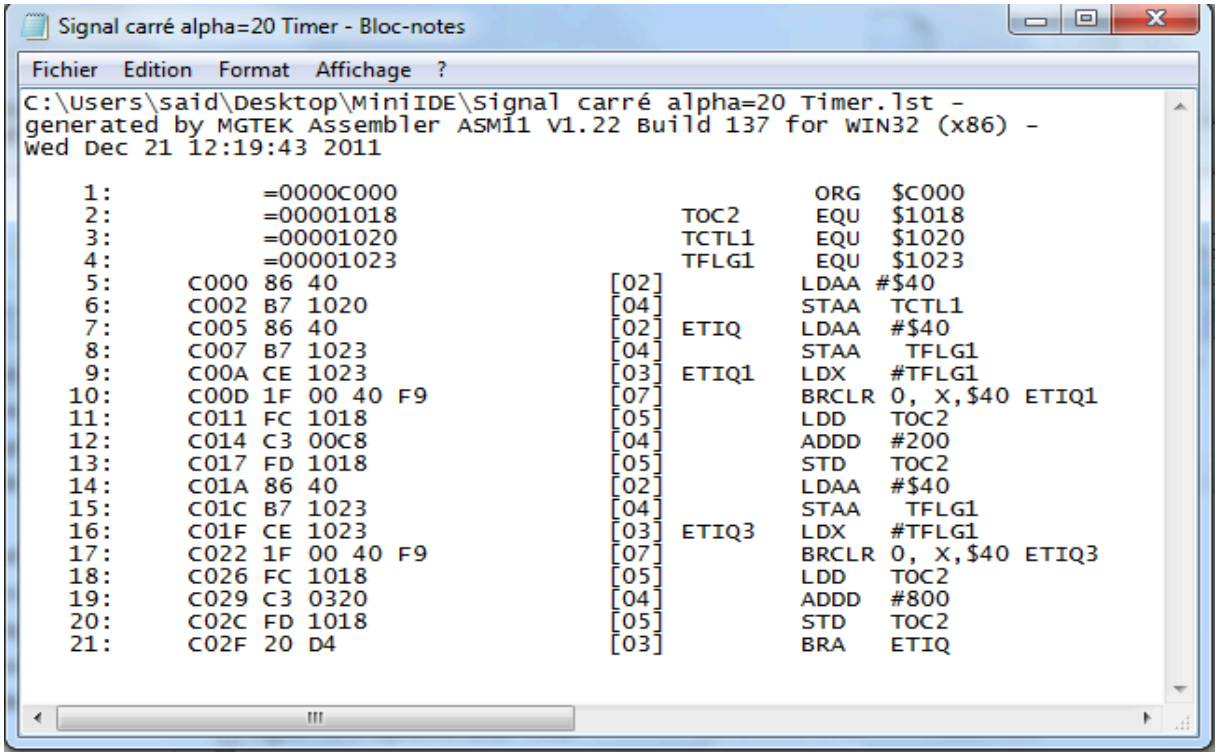


Image2 : *signal carré avec TIMER α=20*.lst

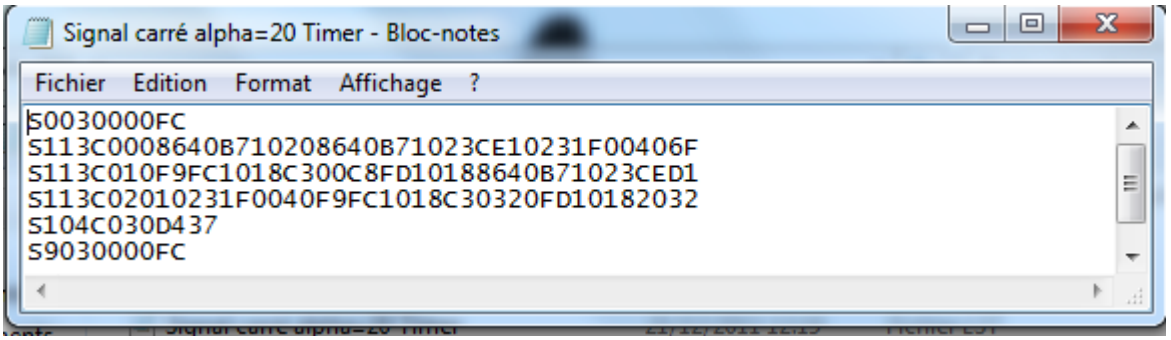


Image3 : *signal carré avec TIMER α=20*.S19

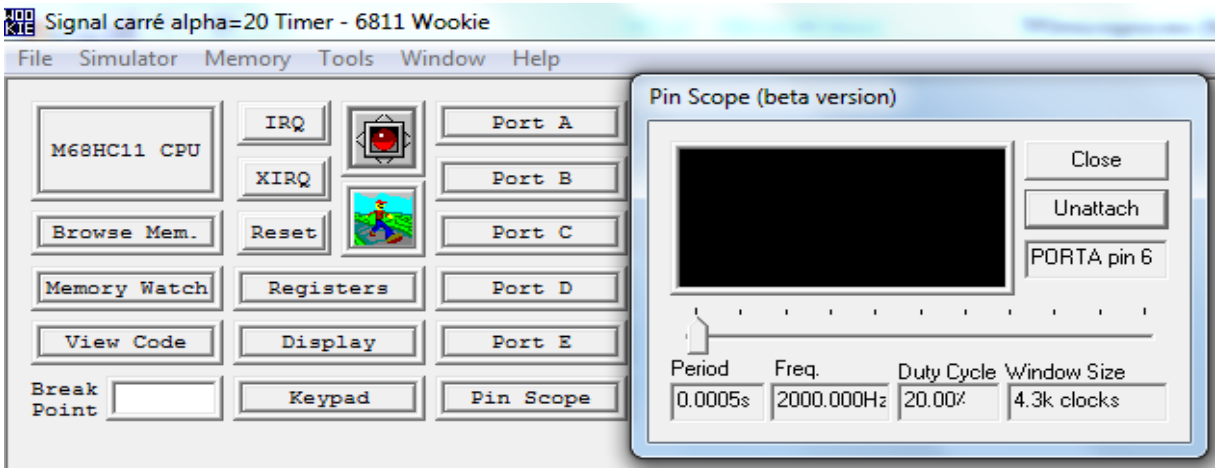
On est maintenant prêt à charger et exécuter notre programme.

2.2. Chargement et exécution du programme :

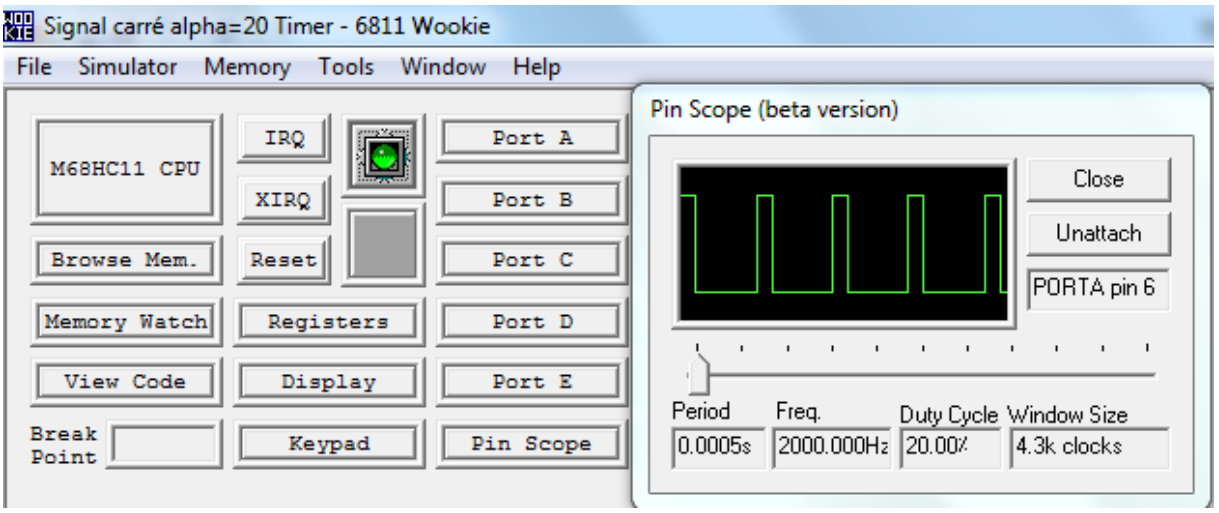
On lance le simulateur « Wookie »

- ✔ On charge le fichier *signal carré avec TIMER α=20*.S19 avec 'Load .S19 file' dans le menu 'File'.
- ✔ Dans la fenêtre 'Set HC11 Mode' on saisit l'adresse de début de programme ($C000)
- ✔ En suite une fenêtre 'Choose LST file format', on choisit 'GCC3 [Addr : Code] et on tape OK.
- ✔ Une fenêtre 'Code View' s'affiche qui contient le fichier .lst du programme.

Avant l'exécution :



Après l'exécution :



Après l'exécution on voit bien que notre programme est bien exécuté.
D'après la visualisation du signal on voit que sa fréquence est de **2000Hz.**

3- Troisième programme :

3-*1 – Ecriture et compilation du programme :*

On saisit le programme dans l'environnement de développement « MiniIDE », puis on l'enregistre sous l'extension .asm 'Exercice3.asm'.

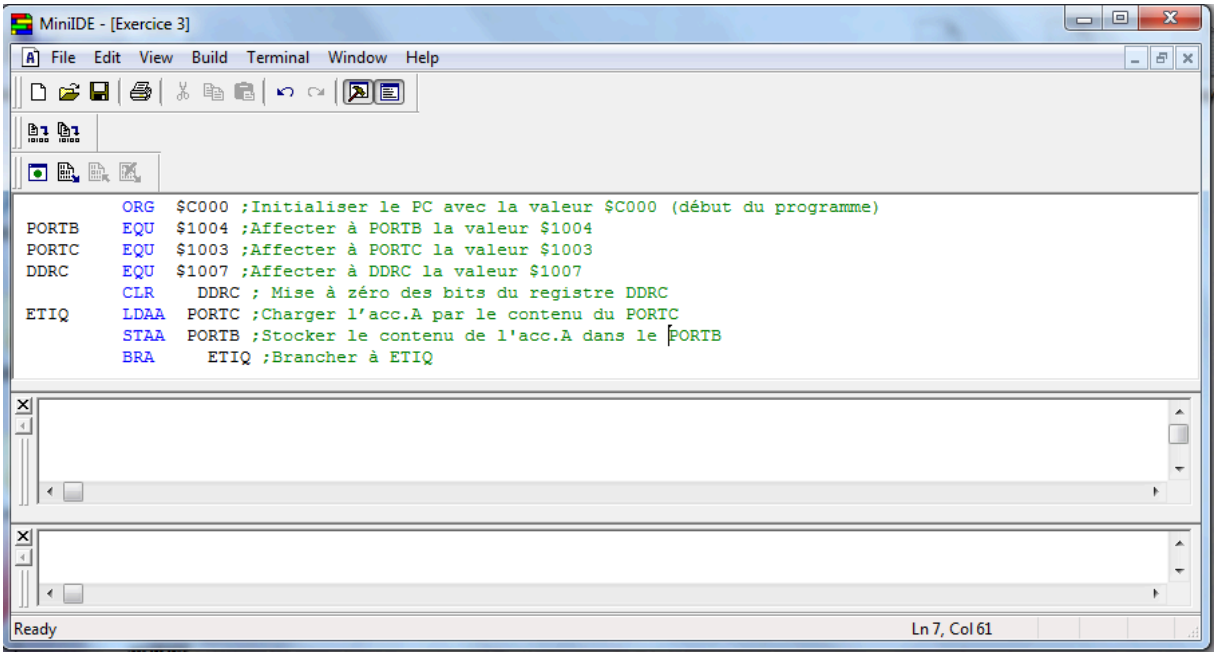


Image1 : Exercice3.asm

✔ On compile le programme avec 'Build Current'

✔ Un message s'affiche dans la fenêtre 'Output Window' indiquant l'emplacement du programme, nombre d'avertissement et d'erreurs.

Une compilation réussite produit deux fichiers :

✔ *Exercice3*.lst

✔ *Exercice3*. S19

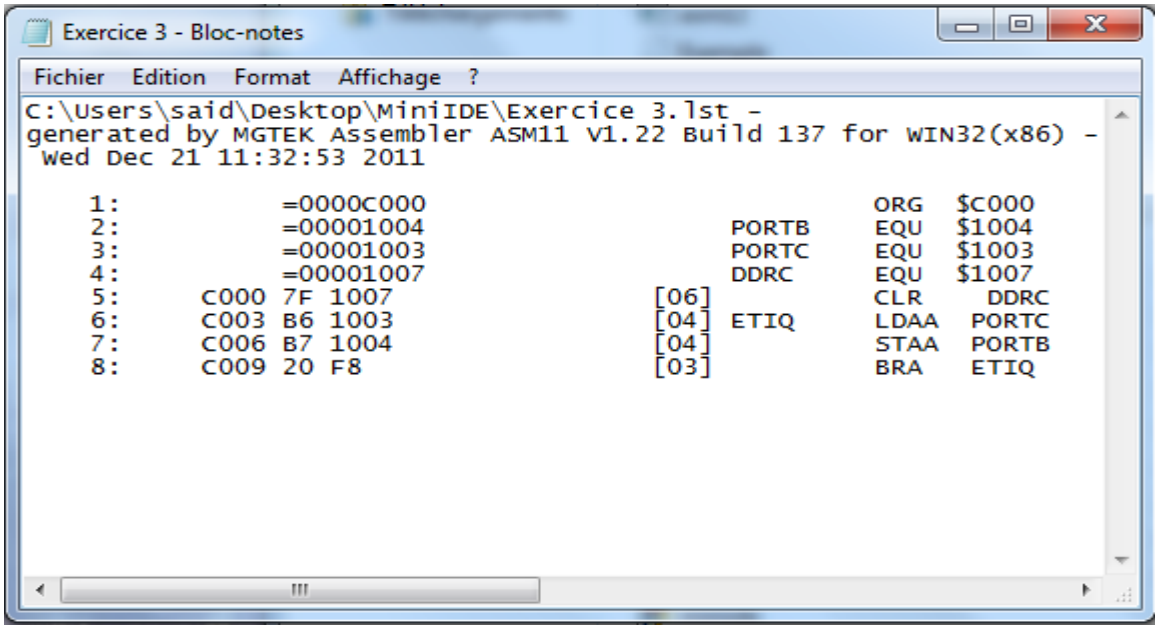


Image2 : *Exercice3*.lst

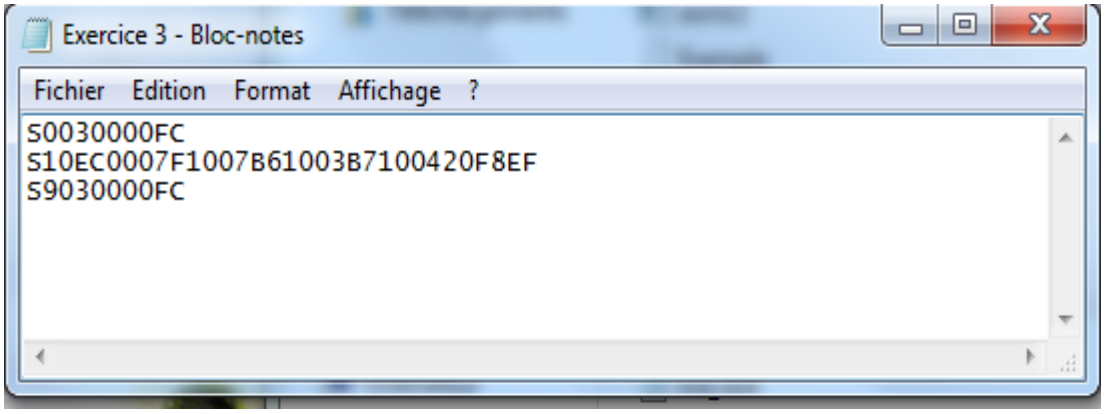


Image3 : *Exercice3*.S19

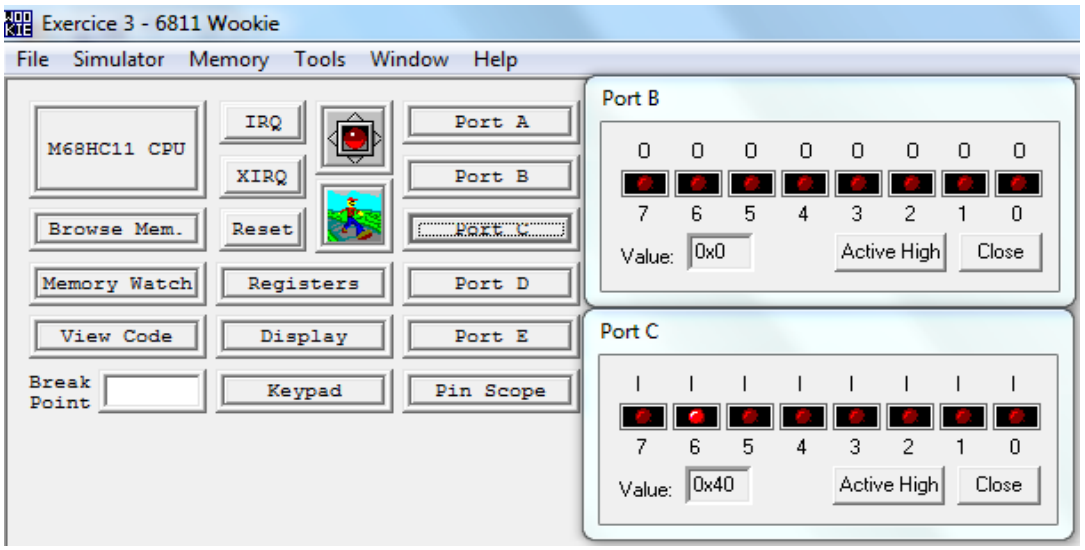
On est maintenant prêt à charger et exécuter notre programme.

2.2. Chargement et exécution du programme :

On lance le simulateur « Wookie »

- ✔ On charge le fichier *Exercice3*.S19 avec 'Load .S19 file' dans le menu 'File'.
- ✔ Dans la fenêtre Set HC11 Mode' on saisit l'adresse de début de programme ($C000)
- ✔ En suite une fenêtre 'Choose LST file format', on choisit 'GCC3 [Addr : Code] et on tape OK.
- ✔ Une fenêtre 'Code View' s'affiche qui contient le fichier .lst du programme.

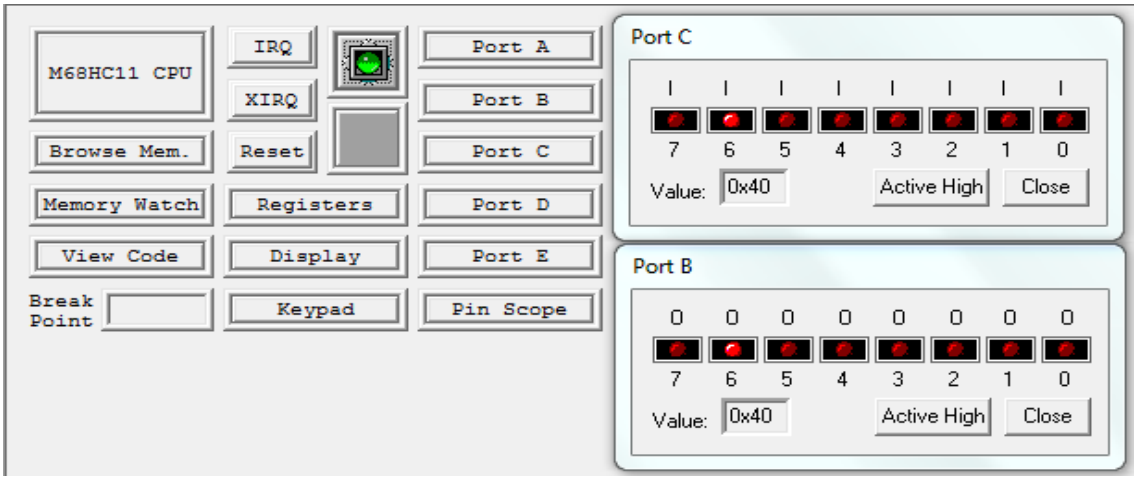
Avant l'exécution :

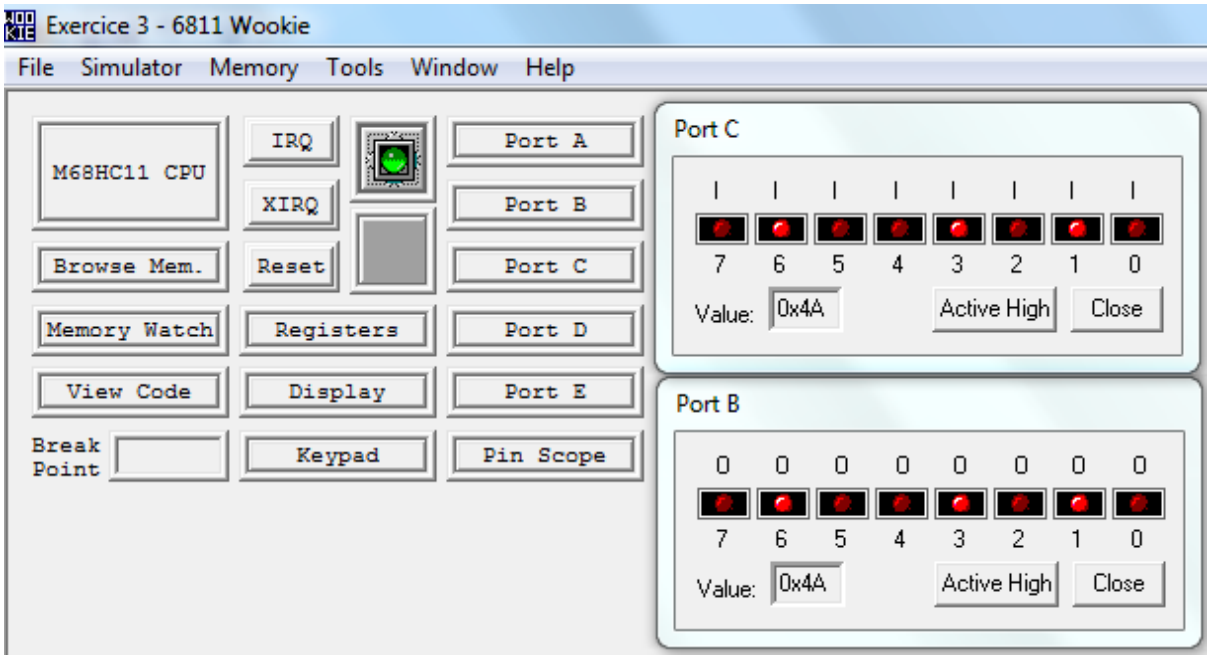


Remarque :

On voit bien que lorsqu'on a chargé le PORTC par une valeur le contenu du PORTB ne change pas, donc le programme n'est pas encore exécuté.

Après l'exécution :





Remarque :

A chaque fois qu'on charge le Port C par une valeur quelconque le Port B affiche la même valeur donc on peut dire que notre programme s'est bien exécuté.

CONCLUSION :

Le troisième TP consiste à développer des programmes en microcontrôleur MC68HC11 en utilisant son jeu des instructions, ses ports et les registres du TIMER comme des outils pour réaliser différents programme(on s'est servi de l'environnement du développement MiniIDE et le logiciel simulateur Wookie) :

-Pour allumage de LEDs (ou encore fonctionnement des LCD…)

- Génération de différents type de signaux avec ou sans utilisation du TIMER (on précise le rapport cyclique et la fréquence désirés …)

- Stocker des valeurs différentes dans un registre unidirectionnel (comme le PORTB) en changeant les valeurs d'un registre bidirectionnel (comme le PORT C)

-… etc.